# 2 Основные сущности и ресурсы Unix-систем

Как уже говорилось, система предоставляет ***пользователю*** свои ресурсы в виде ***процессов*** и ***файлов***.

## 2.1 Пользователи и группы

Пользователь «вообще» – трудноопределимое понятие, т.к. находится вне системы. Система оперирует зарегистрированными пользователями.

***Зарегистрированный пользователь*** (***User***) – фактически ***учетная запись*** (***account***), содержащая характеристики (***атрибуты***) этого пользователя:

***Идентификатор*** пользователя (***User ID***, ***UID***) – числовое значение, традиционно целое положительное.

***Имя*** пользователя (***User name***, ***login***), под которым он известен в системе и входит в нее

***Пароль*** (***Password***) – требуется для аутентификации пользователя при входе в систему, может быть пустым (не всегда)

***Первичная группа*** пользователя (***Primary group***) – в виде идентификатора ***GID***

***Домашний директорий*** (***Home directory*** или просто ***home***) – директорий, который будет текущим сразу после входа пользователя в систему; как правило, пользователь получает полные права на содержимое своего home-директория

«***Shell по умолчанию***» (***Default shell***) – программа, запускаема в качестве ***лидера сеанса*** (см. ниже) после входа пользователя в систему. Обычно это бывает одна из разновидностей shell

Описание пользователей хранится в файле **`/etc/passwd`** (однако, вопреки названию, сами пароли в этом файле отсутствуют – они хранятся в отдельном файле **`shadow`** с намного более ограниченными правами доступа). Формат файла – текстовый CSV (Comma-Separated Values)

Пример строк учетных записей:

```
root:x:0:0:root:/root:/bin/bash
mysql:x:60:114:MySQL database admin:/var/lib/mysql:/bin/false
g653501:x:1035:100::/home/g653501:/bin/bash
g653502:x:1036:100::/home/g653502:/bin/bash
g653503:x:1037:100::/home/g653503:/bin/bash
```

Пояснения:

– Все пользователи имеют непустые пароли (отмечено символом «**x**»)

– Пользователь **mysql** фактически не может войти в систему, т.к. его shell – однократно выполняемая команда (**/bin/false**); он участвует только в управлении правами

– Пользователи **g653501**, **g653502** и **g653503** делят одну и ту же первичную группу (идентификатор группы **100**)

***Группа*** (***Group***) позволяет объединять пользователей, однотипных или схожих по какому-либо признаку, что имеет большое значение при управлении правами.

Характеритики (атрибуты) групп:

***Идентификатор*** группы (***Group ID***, ***GID***)

***Имя*** группы (***Group name***)

***Пароль*** – обычно пустой, но может быть определен для привилегированных групп

***Список*** пользователей группы (***Group list***)

Описание групп, включая и списки включенных в них пользователей – **`/etc/group`**

Пример записей групп:

```
mysql:!:114:
root:x:0:
users:x:100:g052004,g253503,g253504,g253505,\
g353501,g353502,g353503,g353504,g453501,\
g453502,g453503,g453504,g553501,g553502,g553503,\
g653501,g653502,g653503,ibacourses,\
g753501,g753502,g753503,g753504
```

Пояснения:

– Группы **mysql** и **root** (первичные для пользователей **mysql** и **root** соответственно) пока не предусматривают подключения к ним других пользователей (тех пользователей, для которых группа является первичной, перечислять в ней не нужно)

– Группа **users** объединяет много «обычных пользователей» (в данном случае многих из них можно было бы не перечислять)

Специальный привилегированный пользователь («***суперпользователь***») – ***администратор***, традиционное имя **`root`**. Имеет почти неограниченные права на почти любые действия в системе.

Имя **`root`** не обеспечивает никаких привилегий – значение имеет лишь идентификатор **`UID = 0`**.
Скрытие (маскировка) пользователя-администратора под иным, отличным от **`root`**, именем может использоваться как мера (сама по себе слабая) повышения защищенности системы.

В некоторых системах, например семействе Linux Ubuntu, при стандартной инсталляции создается и становится доступным лишь «первичный» пользователь – обладающий почти всеми правами **`root`**, но им не являющийся; «настоящий» же **`root`** «включается» только временно – посредством команды **`sudo`**.

Встречаются системы, где предусмотрен пользователь с правами выше, чем root, вход от имени которого возможен только в специальном однопользовательском режиме – например, **`installator`** в системах DRS 6000, включающийся при запуске машины с помощью механического ключа.

(Аналогия: Windows NT и последующие – неявно создаваемый пользователь **`Administrator`**, регистрируемый при инсталляции первый пользователь, и т.д.)

В дальнейшем будет соблюдаться традиционное именование пользователя-администратора.

## Практическая часть: Некоторые команды

**`who`** – *информация об активных пользователях системы*
**`whoami`** *или* **`who am i`** – *информация о текущем пользователе*

**`useradd`** – *создание нового пользователя*
**`groupadd`** – *создание новой группы*
**`passwd`** – *изменение пароля пользователя*

**`write`** – *текстовое сообщение пользователю*
**`wall`** – *текстовое сообщение всем активным пользователям (можно блокировать)*

**`exit`** – *завершение текущего экземпляра shell, в т.ч. приводящее к завершению сеанса*
**`logout`** – *завершение сеанса*
**`reboot`**, **`halt`**, **`init n`** – *обычно требуют привилегий*

## 2.2 Файловая система, файлы, потоки ввода-вывода

Одна из идеологических концепций Unix – унификация доступа к разнообразным ресурсам посредством файловой системы (соответственно, с единой схемой идентификации и единым набором системных вызовов). Очевидно, необходима возможность представления файлов разнообразного назначения и с разнообразным содержимым.

Типы файлов:

– обычные (***регулярные***) файлы (программы и данные), отдельного обозначения нет

– ***d*** – файлы-***директории*** (каталоги) – файл со списком содержащихся в нем файлов

– ***l*** – файлы-***ссылки*** (символические, ***symbolic***/***soft link***) – файл, указывающий на другой файл

– ***c*** и ***b*** – файлы-***устройства*** (***символьные*** и ***блочные***)

– ***p*** – файлы-***каналы*** (***pipe*** или ***FIFO***) – файлы со специальным режимом доступа, основное назначение – обеспечение обмена данными между процессами

– ***s*** – файлы-***сокеты*** – еще одна разновидность коммуникационного ресурса (здесь имеются в виду сокеты, отображенные на файловую систему, а не сетевые)

Тип файла хранится вместе с другими его характеристиками (см. ниже) и обычно не считается ***атрибутом*** (см. ниже).

***Идентификация*** файлов:
– ***имена*** – в файловой системе
– файловые ***дескрипторы*** (***file descriptor***, ***fd***) – после открытия, используются для ввода-вывода в программах

***Потоки ввода-вывода*** (***I/O stream***) – абстракция для унифицированного представления передачи данных. Как поток можно рассматривать любой открытый файл, но чаще подразумевается файл или устройство ***последовательного*** доступа.

***Имена*** файлов – регистрочувствительные, ограничение длины зависит от файловой системы, допустимое множество символов – почти все, кроме «`/`» и «`\0`» (но многие теоретически «разрешенные» создают серьезные проблемы и реально не используются)[1].

Символ «точка» («`.`») используется для визуального разделения имени файла, но это не имеет значения для системы. Последнюю отделенную точкой часть имени принято называть «***суффиксом***», и многие Unix-программы суффиксы никаким специальным образом не учитывают и не обрабатывают. Однако имена, начинающиеся с точки, считаются «невидимыми» – по умолчанию они не показываются (но будут показаны при явном требовании этого).

[1] Утверждают, что С.Борн, автор классического командного интерпретатора sh, тестировал его в том числе на каталоге, содержащем 254 программно сгенерированных файла с односимвольными именами – таким образом были использованы все символы, кроме двух запрещенных.

Файловая система Unix – ***иерархическая*** (древовидная). Это же унаследовано большинством современных файловых систем.

Верхний уровень иерархии директориев – ***корневой*** директорий (root, не имеет отношения к пользователю-администратору), обозначаемый символом «/». Все остальные – его ***поддиректории***, которые, в свою очередь, также могут иметь (и обычно имеют) поддиректории более низкого уровня.

***Полное*** имя файла – имя вместе с ***путём***, т.е. указанием директория и всех вышестоящих директориев. Пути различаются:
– ***относительные*** – отсчитываются от текущего директория и начинаются непосредственно с его имени (или состоят только из имени файла)
– ***абсолютные*** (***полные***) – отсчитываются от корневого директория, запись начинается с «/»

Дополнительно используются «особые» имена, реально содержащиеся в каждом директории, кроме корневого (по описанной выше причине они «невидимы»):

– « . » – текущий директорий
– « . . » – родительский (вышестоящий) директорий

а также принятое обозначение (не имеет реального отражения в файловой системе, но «расшифровывается» программами при обращении по путям):

– «~» – домашний директорий текущего пользователя

Некоторые основные директории (имена можно считать стабильными в различных семействах ОС):

**`/bin`**, **`/sbin`** – исполняемые файлы (в общем случае не обязательно именно двоичные)

**`/lib`** – библиотеки языка С и других, используемые исполняемыми файлами

**`/dev`** – файлы устройств

**`/mnt`** – данные для монтирования устройств в единую файловую систему

**`/etc`** – конфигурационные (настроечные) и аналогичные по назначению файлы (и утилиты)

**`/var`**, **`/tmp`** – «рабочие» файлы программ и временные файлы

**/usr** – директорий «дополнительно установленных» «пользовательских» программ (условно):

**/usr/bin**, **/usr/lib** – исполняемые файлы и библиотеки

**/usr/etc** – конфигурационные файлы

**/usr/share** – разделяемые (совместно используемые) файлы

и т.д.

**/home** (при необходимости также **/home1**, **/home2**, ...) – домашние директории пользователей

**/lost+found** – файлы и директории, «потерянные» в результате сбоев и найденные при проверке файловой системы

и т.д.

Основа представления файлов – структура ***inode*** («индексный дескриптор»), связывающая имя файла, его метаданные («данные о данных»: атрибуты и информация о размещении) и, далее, собственно данные (содержимое).
Все inode идентифицируются номерами, уникальными в пределах одного устройства (одной файловой системы).

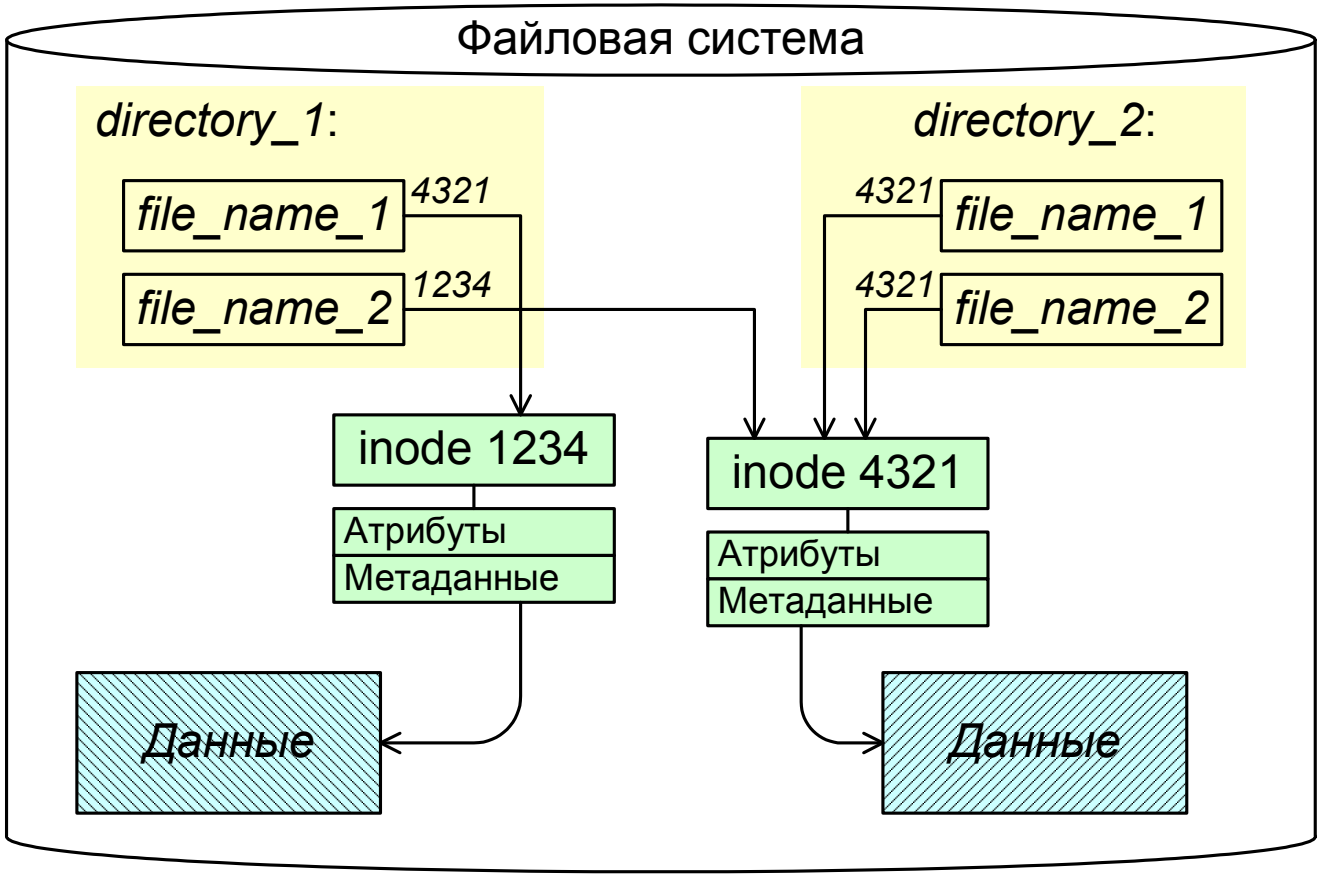


Рис. – Файлы, inode, метаданные

В приведенном примере один из файлов (inode=1234) известен в файловой системе под единственным именем, второй (inode 4321) – под несколькими именами в нескольких директориях.

Особенность файловых систем Unix – имя файла в inode не включается, оно присутствует только в директориях. Как следствие, один и тот же файл может фигурировать под разными именами и/или в разных директориях, при этом содержимое файла не совпадает, а действительно является одним и тем же (одни и те же метаданные ссылаются на одни и те же блоки диска), т.е. разные имена являются синонимами. Поэтому имя, вообще говоря, является просто ***ссылкой*** (***link***) на inode, и таких ссылок может быть несколько. Эти ссылки принято называть ***жесткими*** (***hard link***), в отличие от символических. Очевидно, любые изменения в файле, доступном по одному из своих имен, становятся видны и под всеми другим жесткими ссылками. Файл существует до тех пор, пока есть хотя бы одна жесткая ссылка на него.

Также очевидно, что жесткие ссылки не могут указывать на файл в физически другой файловой системе, пусть и смонтированной в единую иерархию. Сделать это позволяет только символическая ссылка – фактически просто имя файла с полным путём к нему.

Другие характеристики файлов:

***Владелец*** (***пользователь***) файла. Предусмотрено сразу два «владельца»:
– владелец – конкретный пользователь
– владелец – группа

Таким образом, файл (объект) принадлежит одновременно и независимо некоторому пользователю и одновременно некоторой группе, в общем не обязательно включающей этого пользователя. Это активно используется при управлении правами доступа.

Дата и время (***timestamp***):
– создания файла
– последней модификации файла
– последнего обращения к файлу

Точный ***размер*** файла (в байтах) – необходим в дополнение к метаданным, так как последний блок файла может быть заполнен частично. (В ряде файловых систем предусмотрено разделение этого блока между несколькими файлами.) Более того, обычно допустимы «дырявые» файлы: заявленный размер может превышать суммарный объем выделенных блоков, и часть файла («дыры») остаётся без реально выделенного дискового пространства; блок (возможно, со своим текущим содержимым!) будет выделен при первом реальном обращении к такой «дыре».

***Атрибуты*** файлов – характеристики, отвечающие за те или иные особенности и свойства объекта файловой системы. К основным атрибутам относятся биты режима (права доступа), биты SUID и SGID (также управляют правами для исполняемых файлов), и Sticky bit.

***Биты режима*** определяют ***права доступа***: какие действия в отношении объекта разрешены для различных пользователей.

Права доступа играют важную роль в любой многопользовательской системе, управление ими составляет значительную часть задач администрирования. В Unix система прав базируется на трёх видах доступа: ***r***ead – чтение, ***w***rite – запись/модификация, e***x***ecute – исполнение
и трёх категориях пользователей: ***u***ser – пользователь/владелец, ***g***roup – группа, ***o***thers – прочие пользователи, не являющиеся владельцами.

Таким образом, права («режим») доступа кодируется тремя тройками битов (флагов), которые традиционно обозначаются восьмеричными цифрами. Установленный флаг (единичное значение бита) соответствует наличию разрешения, но интерпретация зависит и от объекта:

| Бит | Предоставляемые права | |
|---|---|---|
| | **Файл** | **Директорий** |
| **R** | Чтение содержимого | Чтение содержимого (списка файлов) |
| **W** | Запись в файл, модификация содержимого | Изменение содержимого (добавление и удаление файлов) |
| **X** | Исполнение файла (двоичный исполняемый код либо текстовый скрипт) | Переход в директорий (сделать директорий текущим). Обычно также необходимо и для чтения списка файлов |

Для файлов – символических ссылок часто устанавливаются полные права, так как они будут распространяться только на саму ссылку, но не на содержимое доступного через нее файла (которому соответствует другой inode). Однако ограничение прав может иметь смысл и для ссылки, например для предотвращения её несанкционированного изменения.

Принцип применения прав – «суммирующий»: если действие разрешено для хотя бы одной категории пользователей, к которой относится инициатор действия, то действие разрешено.

Порядок проверки прав:

1) Пользователь – суперпользователь (root)? Если да, то действие разрешено независимо от битов режима. (Исключение: файл, не имеющий ни одного установленного бита «**x**», не может быть выполнен также и суперпользователем, так как, вероятно, не является исполняемым.)

2) Пользователь – владелец файла? Если да, и установлен соответствующий флаг в тройке «user», то действие разрешено

3) Пользователь – член группы – владельца файла? Если да, и установлен соответствующий флаг в тройке «group», то действие разрешено

4) Если установлен соответствующий флаг в тройке «others», то действие разрешено

5) Действие не разрешено (not permitted).

Права доступа к файлу и к содержащему его директорию не зависят друг от друга. Если доступ разрешен для файла, но запрещен для директория, файл будет доступен, но только с указанием пути к нему, не пытаясь войти в директорий. Запрет доступа к файлу не мешает видеть его в списке или удалить.

Аналогично действуют права для других объектов, представимых как файлы и доступных посредством дескрипторов **`fd`**, а также, например, для объектов System V IPC. В целом схема управления правами в Unix-системах менее гибкая, чем, например, в Win NT, но она достаточна для большинства ситуаций, при этом проще и экономнее.

## Другие атрибуты

Биты ***смены идентификатора*** пользователя ***SUID*** (Set User ID) и группы ***SGID*** (Set Group ID) – для исполняемого файла «эффективным пользователем» (см. ниже) будет считаться не текущий пользователь – инициатор выполнения программы, а владелец исполняемого файла (для группы поведение аналогично). Позволяет непривилегированному пользователю обращаться к программам, требующим специальных прав.

***Sticky bit*** – «бит сохранения задачи», первоначально предназначался для оптимизации работы с исполняемыми файлами – удержание («залипание») загруженного файла в памяти после завершения порожденного из него процесса. В настоящее время чаще применяется к директориям – разрешение удаления файлов только их владельцами независимо от прав доступа иных пользователей.

## Дополнительные атрибуты

(не обязательно поддерживаются всеми файловыми системами)

– No Access Time – не модифицировать время последнего обращения к файлу

– Append Only – только дописывание (для файла), добавление файлов (для директория)

– No Dump – не включать в резервную копию файловой системы

– Immutable – запрет любых изменений

– Secure Deletion – «безопасное» удаление с очисткой принадлежавших файлу блоков

– Synchronouse Mode – «сквозная» (без буферизации) синхронная запись в файл

## Практическая часть: Некоторые команды

Информация о файлах и их характеристиках:

**`ls, ls`** ***`my_dir`*** – *содержимое директория (только имена)*
**`ls -l`** – *то же с подробными сведениями о файлах: права, атрибуты, владелец, размер*
**`ls`** ***`my_file.*`*** – *отображать только файлы, соответствующие шаблону*
**`ls .*`** – *отображать все файлы, включая «невидимые»*
**`ls -d`** – *для поддиректориев отображать их имена, а не содержимое*

Команды навигации и простые манипуляции в файловой системе:

**cd *my_dir*** – *смена текущего директория (вход в директорий)*
**pwd** – *имя текущего директория*
**mkdir *new_dir*** – *создание нового директория*

**cp *my_file_pathname new_file_pathname*** – *копирование файла*
**mv *my_file_pathname new_file_pathname*** – *перемещение или переименование*
**mknod** – *создание объекта файловой системы (разнообразные параметры)*

**ln *имя_существующего_файла имя_синоним*** – *создание жесткой ссылки*
**ln -s *имя_существующего_файла имя_ссылка*** – *создание символической ссылки*

**mount *device mounting_point*** – *монтирование устройства в файловую систему*
**umount *device***, **umount *mounting_point*** – *размонтирование*

**rm *any_file_name*** – *удаление файла, директория или иного объекта*
**rmdir *dir_name*** – *«безопасное» удаление директория (только пустого)*

Поиск:

**find** – *поиск в файловой системе (разнообразные параметры)*
*например:*
**find *нач_директорий* -name *шаблон_имени* -print** – *поиск по имени, начиная с заданного стартового директория, результаты отображать в* **stdout**

**grep *шаблон_строки список_файлов_или_шаблон_имён*** *– поиск строки в файлах*

Права, владельцы, атрибуты:

**chmod 777 *file1 file2* ...**, **chmod 640 ...**, **chmod 751 ...** *и т.п. – задание прав в форме кода*

**chmod a+rwx ...**, **chmod ug+x ...**, **chmod o-rw ...** *и т.п. – изменение прав в символической форме*

**`chown`** ***`new_owner_user file1 file2 ...`*** – *смена пользователя*

**`chgrp`** ***`new_owner_group file1 file2 ...`*** – *смена пользователя-группы*

**`lsattr`** – *атрибуты файла (поддержка зависит от ОС и файловой системы)*

**`chattr`** – *изменение атрибутов файла (поддержка зависит от ОС и файловой системы)*

**`touch`** – *изменение временных меток файла (имитация доступа), при необходимости создание файла*

Элементарный ввод-вывод:

**`echo`** ***`строка строка . . .`*** *– вывод строк в stdout, в т.ч., например:*
**`echo`** ***`строка > файл`*** *– простейшее создание короткого файла*

**`read`** ***`имя_переменной`*** *– чтение строки из stdin в переменную*

## 2.3 Процессы

***Процесс*** (вычислительный процесс, ***process***) – динамический системный объект, соответствующий выполняющейся программе, экземпляр выполняющейся программы.

***Поток*** (вычислительный поток, ***thread***) – определение аналогично «общетеоретическому»: последовательность машинных команд, выполняемых параллельно и независимо по отношению к другим таким же последовательностям команд. Технически в большинстве Unix-систем реализуется как «облегченный» процесс.

По историческим причинам большее внимание в Unix-системах уделяется процессам. Потоки будут более подробно рассмотрены как часть подсистемы управления процессами вместе с соответствующим системным API.

Процесс состоит из кода (машинных команд, инструкций), данных (пространство адресов и пространство дескрипторов) и «метаданных» – сведений о самом процессе. Код и данные – ***образ*** процесса, обычно организован как несколько ***сегментов***. Кроме образа, с процессом связаны ряд ***атрибутов***, ***переменные окружения*** и ***контекст выполнения***.

Если программа – не двоичный исполняемый файл, а текстовый (скрипт), то процесс – экземпляр интерпретатора этого скрипта.

Процессы проходят ***жизненный цикл*** из сменяющих друг друга ***состояний***, могут порождать другие процессы и взаимодействовать между собой и с системой посредством ***API***.

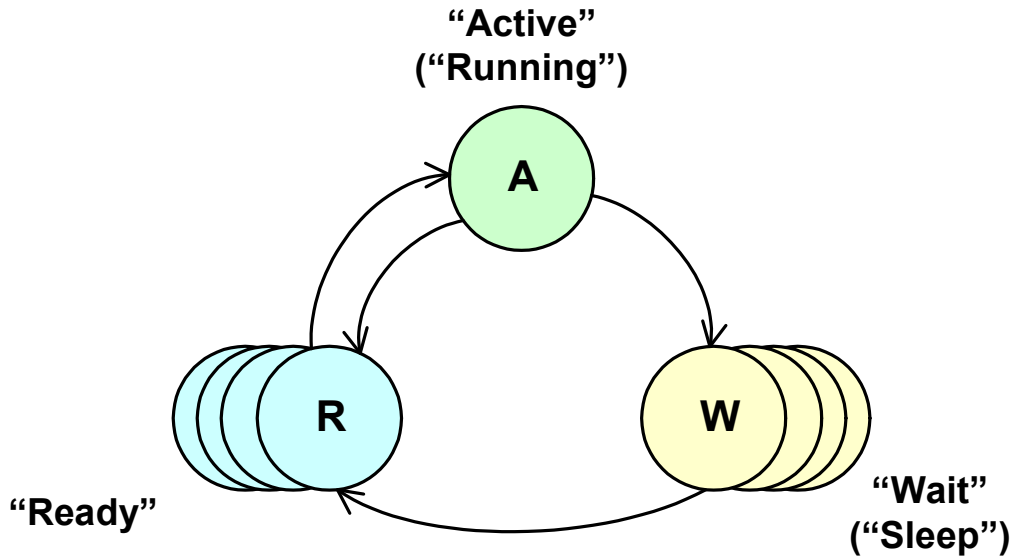


Рис. – Состояния процесса (потока)

Виды процессов:

– ***Системные*** (привилегированные; **`init`** и др.)

– ***Неинтерактивные*** или ***демоны*** (аналог резидентных программ и служб) – непривилегированные, выполняются независимо от текущего пользователя, не входят в текущий сеанс пользователя и не имеют доступа к терминалу (см. ниже).

– ***Интерактивные фоновые*** – обычные процессы пользователя в текущем сеансе, не владеющие в данный момент «фокусом ввода» консоли

– ***Активный*** интерактивный процесс – единственный в сеансе процесс, получающий пользовательский ввод из консоли.

При использовании оконного интерфейса (***desktop***) деление процессов на активный и фоновые теряет актуальность, но понятие «фокуса ввода» сохраняется (ввод с клавиатуры попадает только в одно «сфокусированное» окно).

***Идентификация*** процессов – уникальные в системе ***Process ID*** (***PID***).
Целое число, генерируется как счетчик, по достижении максимума «заворачивается» на повторный проход, пропуская занятые значения. Собственный PID доступен для процесса, но не может быть им изменен, и вообще не меняется в течение всего жизненного цикла процесса.

Другие атрибуты:

«***Родительский***» PID (***parent PID***) – ***PPID***.
PPID определен всегда и для каждого процесса, и всегда валиден. PPID доступен для процесса, но не может быть им изменен, зато изменяется системой при исчезновении (завершении) процесса-родителя – в этом случае он меняется на PID системного процесса, отвечающего за другие процессы (обычно это процесс **`init`**).

Процесс-родитель (в том числе **`init`**) должен контролировать состояние порожденных процессов и не допускать превращение их в «зомби» (zombie).
Процесс (общий «приемный родитель» всех процессов-«сирот» обеспечивает такое поведение.

***Идентификаторы*** пользователя и группы:

«***реальные***» (***RID***, ***RGID***) – сохраняют ID пользователя и пользователя-группы – инициатора выполнения процесса (фактически процесса-родителя), не могут быть изменены

«***эффективные***» (***EUID***, ***EGID***) – применяются для определения прав доступа к объектам, устанавливаются равными UID и GID исполняемого файла, если у того есть атрибуты SUID, UGID соответственно, иначе равны RID, GID.

Табл. – Назначение и использование RID, EUID, RGID, EGID

***Приоритет*** (***nice number***) – числовой показатель «любезности» поведения процесса по отношению к другим процессам. Значение изменяется от **`-20`** до **`19`**, причем меньшее значение соответствует большему приоритету.

Обычный пользователь может только увеличить свой nice number, то есть понизить приоритет.

Nice number не обязательно соответствует (точнее, обычно не соответствует) значениям приоритетов, которыми оперирует ядро конкретной системы. Это – часть её стандартизованного «внешнего» интерфейса.

***Терминал*** (***TTY***) – имя устройства в директории **`/dev/`**, ассоциированного с физическим терминалом, виртуальной локальной или удаленной консолью, с которой связан ввод-вывод процесса.

Неинтерактивные процессы (демоны) не связаны ни с каким терминалом.

Идентификатор ***сеанса*** (session ID, ***SID***) – PID лидера сеанса (см. ниже), к которому относится процесс. Очевидно, совпадает с собственным PID, если процесс сам является лидером. Процесс может самостоятельно создать сеанс, лидером которого станет он же.

Также с процессом связана статистика его работы: время, затраченное в ***режиме ядра*** и в ***режиме задачи***.

***Переменные окружения*** (***переменные среды***, ***блок окружения***, ***environment***) – набор именованный строковых переменных, определенных для каждого процесса. Ряд имен переменных стандартизован, другие создаются процессами произвольно. Порождаемый процесс наследует окружение родителя или получает его модификацию или заново созданный набор.

Более подробно рассматриваются в теме командных интерпретаторов.

***Сеанс*** (***session***) – обычно все процессы, выполняемые пользователем от момента его входа в систему до выхода. Кроме того, сеансы могут быть созданы искусственно, отдельными процессами.

***Лидер сеанса*** – процесс, создаваемый первым для пользователя при создании сеанса (либо процесс, целенаправленно создавший новый сеанс).

Таким образом, сеанс составляют лидер сеанса и все процессы, порожденные им непосредственно или другим порожденными им процессами. При завершении лидера завершаются и все процессы этого сеанса.

В качестве лидера сеанса обычно выступает командный интерпретатор «по умолчанию» (default shell), но это может быть и любой другой процесс: указанный в учетной записи пользователя или создавший собственный сеанс

Пример: Специальные пользователи **`halter`** и **`rebooter`**, обеспечивающие соответственно выключение и перезапуск системы. Реализация: в качестве default shell назначены команды halt и reboot с соответствующей настройкой прав их выполнения. Результат: рядовой пользователь не может прекратить работу сервера со своего рабочего места, права суперпользователя рядовым пользователям не предоставлены, но есть возможность воспользоваться указанными пользователями-«функциями» (их пароли известны) с управляющей консоли. (Однако обычно схожего результата добиваются использованием команды **`sudo`**.)

Процессы-демоны создают собственные сеансы и являются их лидерами.

***Порождение*** процессов:

***fork*** (системный вызов **`fork()`**) – создание копии (клона) процесса-родителя по инициативе родителя

***exec*** (системный вызов и функции **`exec**()`**) – замещение процесса-родителя образом из исполняемого файла (тоже инициатива родителя)

Связка ***fork-exec*** – типовое решение для создания новых процессов (не клонов родительских).

Более подробно будет рассмотрено как часть подсистемы управления процессами вместе с соответствующим системным API.

***Завершение*** процессов:

– По инициативе самого процесса – выполнение (явно или неявно) системного вызова **`exit()`**

– Принудительно по сигналу (см. ниже) по инициативе другого процесса или в результате аварии (ошибки) – фактически по инициативе системного процесса, контролирующего такие ошибки.

Завершившийся процесс переходит в состояние «зомби» (zombie), и его родителю отправляется сигнал **`SIGCHLD`** (см. ниже). Процесс остается «зомби», пока этот сигнал не будет обработан.

***Код завершения*** (***код возврата***) – целочисленное (обычно предполагается неотрицательным) значение, возвращаемое процессом при его завершении и доступное в дальнейшем как часть информации о процессе и как переменная окружения родителя.

Традиционно нулевое значение интерпретируется как признак успешного завершение, ненулевое – как признак ошибки. В случае аварийного (по сигналу) завершения процесса код возврата имеет значение (**`128 + №_сигнала`**)

Один из основных механизмов управления процессами и их взаимодействия – ***сигналы*** (разновидность сигнальных IPC). Сигналы характеризуются только типом/номером (номерам соответствуют общепринятые символические имена), направляется адресно к определенному процессу или нескольким процессам, реакция на сигнал зависит от его типа, причем обработка большинства сигналов может быть перехвачена самим процессом.

Номера сигналов стандартизованы, но некоторые из них могут различаться в разных системах.

Например:
**1** (**SIGHUP**) – «обрыв линии» («hang up»), отключение терминала, прекращение сеанса; для демонов – требование ре-инициализации
**15** (**SIGTERM**) – сигнал завершения, обычная реакция по умолчанию – завершение процесса, но, например, shell игнорирует этот сигнал, если является лидером сеанса
**20**, **17** или **18** (**SIGCHLD**) – завершение порожденного процесса (для процесса-родителя)

Два сигнала не могут быть перехвачены и переопределены:
**9** (**SIGKILL**) – безусловное завершение
**17**, **19** или **23** (**SIGSTOP**) – останов процесса

Более подробно будут рассмотрены как часть подсистемы управления процессами вместе с соответствующим системным API.

***Стандартные*** потоки ввода-вывода – изначально существуют у каждого процесса и доступны посредством предопределенных заранее открытых дескрипторов. В Unix-системах их три:

| Поток | fd | Описание | Направление | Ассоциирован |
|---|---|---|---|---|
| **`stdin`** | **`0`** | стандартный поток ввода – данные | ввод | консоль – дисплей |
| **`stdout`** | **`1`** | стандартный поток вывода – данные | вывод | консоль – клавиатура |
| **`stderr`** | **`2`** | стандартный поток ошибок – сообщения | вывод | консоль – дисплей |

Как видно, содержимое потоков **`stdout`** и **`stderr`** смешивается при выводе на консоль (по умолчанию), но может быть разделено в случае перенаправления.

Потоки могут быть ***перенаправлены*** – в реальный файл или из файла – или связаны с другим поток, в том числе с потоком другого процесса – ***туннелирование*** данных через ***канал*** (транспортёр). Это дает удобный унифицированный способ организации взаимодействия нескольких процессов (подробнее рассматривается как часть функциональных возможностей командных интерпретаторов).

***Фильтры*** – процессы, изначально рассчитанные на получение входных данных из **`stdin`** и вывод результатов в **`stdout`**, и взаимодействие, таким образом, с другими процессами. Использование явно открываемых файлов при этом также обычно не исключается.

Примеры процессов-фильтров: **`more`**, **`less`**, **`head`**, **`tail`**, **`sort`**, **`shuf`**, **`od`** и т.д. Ряд более сложных программ использует тот же подход и тоже может считаться фильтрами: **`grep`**, **`sed`**, **`awk`** и т.д.

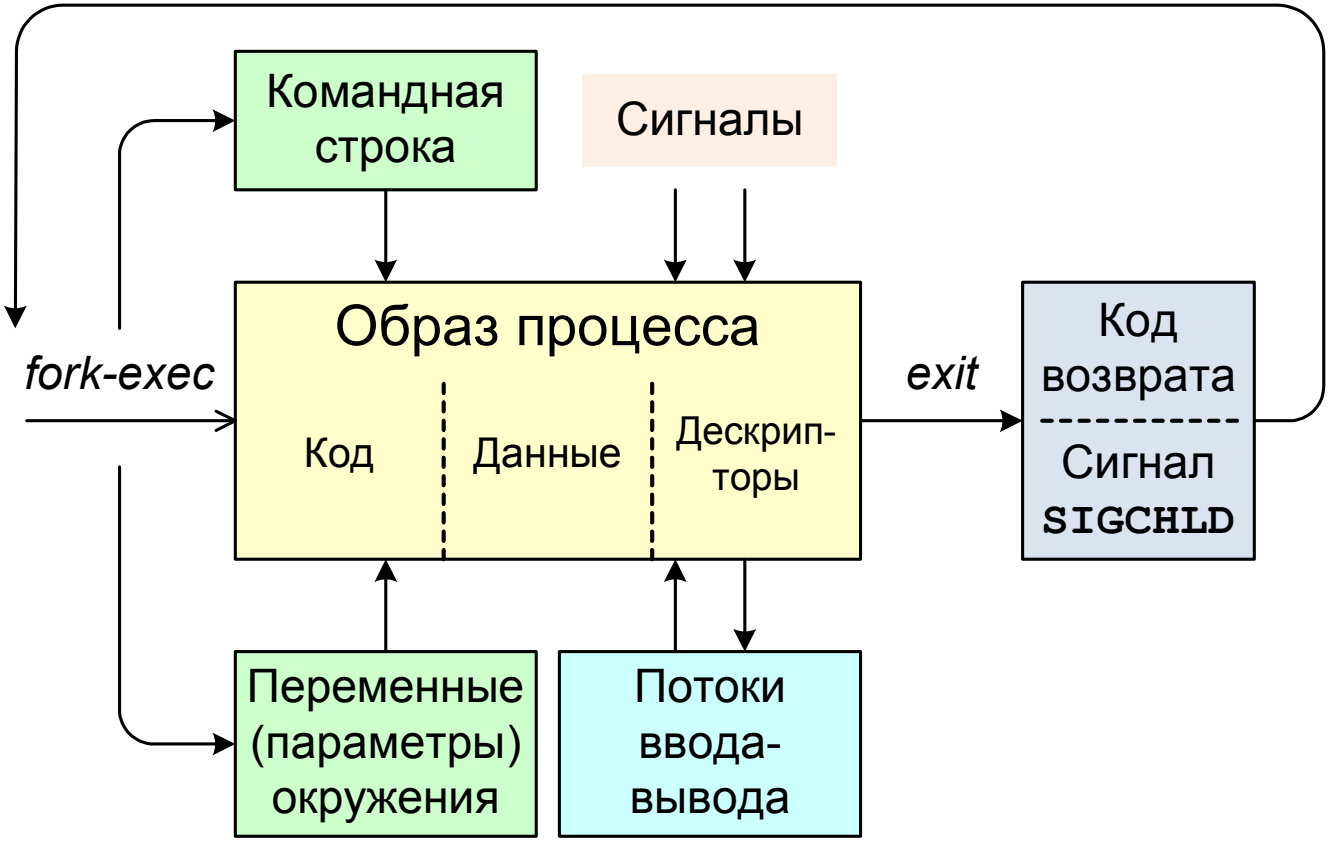


Рис. – Выполняющаяся программа (процесс)

(Схематично. Атрибуты процесса и ряд других подробностей условно не показаны.)

## Практическая часть: Некоторые команды

**`ps`** – список процессов

**`top`** – список процессов, отсортированный по потребляемым ими ресурсам и периодически обновляемый

**`nice`** – управление приоритетом

**`kill`** – посылка сигнала процессу (процессам)

**`... &`** – выполнение команды (процесса) в фоновом режиме

**`fg`** – перевод фонового процесса активный режим, т.е. «на передний план» (foreground)

**`nohup`** – выполнение команды (процесса) без прерывания в случае завершения сеанса

**`time`** – выполнение команды (процесса) с подсчетом статистики времени выполнения.

Более подробно – далее в соответствующих специализированных темах